

网格中流水式计算的一种任务指派算法

王庆江, 桂小林, 郑守淇

(西安交通大学电子与信息工程学院, 710049, 西安)

摘要: 为取得网格中流水式计算的高吞吐率, 提出一种任务指派算法 $X\text{-max-min}$. 在一个流水线中, 任务彼此是并行的, 且每个任务本身是可并行化的. 当多个任务被指派到同一个并行系统时, 通过最小化任务计算成本的最大值确定每个任务分得处理机的个数. 任务用于收发数据集的通信成本依赖其他任务的指派, 故当相关任务的指派未完成时, 需要在任务通信成本中引入均值估计. 任务响应时间是计算成本和通信成本之和, 它是任务指派的函数. 用 max-min 算法确定任务指派, 可有效降低任务响应时间的最大值, 从而使流水线的吞吐率得到提高. 仿真实验表明, $X\text{-max-min}$ 算法使流水线取得的吞吐率与复杂的 Taura 算法相当.

关键词: 网格; 流水式计算; 流水线; 任务指派; 吞吐率

中图分类号: TP393 **文献标识码:** A **文章编号:** 0253-987X(2004)04-0417-03

Task Assignment Algorithm for Pipelined Computing in Grid

Wang Qingjiang, Gui Xiaolin, Zheng Shouqi

(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: To obtain high throughput of the pipelined computation in grid, a task assignment algorithm called $X\text{-max-min}$ was proposed. In a pipeline, tasks were parallel with each other, and each task was parallelizable itself. When more than one task was assigned to an identical parallel system, the number of processors allocated to each task was determined by minimizing the maximum of task computation costs. The task communication cost used for receiving and sending a data set depended on the assignments of other tasks, so the mean estimate was needed to be introduced into the task communication cost when the assignments of related tasks were not completed yet. The task response-time was the sum of the computation cost and communication cost, and was a function of task assignments. Using max-min algorithm to determine task assignments, the maximum of the task response-times could be effectively reduced so that the pipeline throughput was increased. Simulation results show that the $X\text{-max-min}$ algorithm makes pipelines obtain high throughput that is almost the same as the one provided by complicated Taura's algorithm.

Key words: grid; pipelined computing; pipeline; task assignment; throughput

在数字信号处理、图像处理、计算机视觉等领域, 连续产生的数据被分成多个数据集, 数据集处理有延迟时间和吞吐率上的限制, 故数据集处理一般实现为任务流水线, 每个任务负责一个处理环节.

文献[1]假设任务通信成本可忽略或可将通信成本归约到计算成本, 提出了在一个并行机内用串行结构流水线实现吞吐率限制下的最短响应时间调

度算法. 文献[2]支持任务通信成本不可忽略的情况, 通过建立任务响应时间函数, 提出了在一个并行机内提高串行结构流水线吞吐率的任务指派算法. 网格汇聚了许多高性能资源, 流水式计算是网格中的一种典型应用^[3]. 对于以桌面或工作站作为计算资源的异构环境, 文献[4]提出了一种取得流水线高吞吐率的任务指派算法, 该算法向任意序列中的主

收稿日期: 2003-07-09. 作者简介: 王庆江(1968~), 男, 博士生; 郑守淇(联系人), 男, 教授, 博士生导师. 基金项目: 国家高技术研究发展规划资助项目(2001AA111081); 国家自然科学基金资助项目(60273085).

机指派任务,而指派哪些任务和多少任务取决于任务排序,首次任务指派后的流水线吞吐率往往不理想,还需反复多次改派吞吐率瓶颈涉及的那些任务,以进一步提高吞吐率。

假设网格资源以并行系统为主,流水线任务都是可并行化的,则指派一个任务包括指派到哪个并行系统和分得多少处理机;流水线任务之间是并行的,而任一处理机不参加多个任务的运行。据此,本文提出了一种任务指派算法 X-max-min,它可用于网格中的串行结构流水线,以取得高吞吐率。

1 问题定义

假设网格中包含 n 个并行系统,网格的计算资源可表示为 $G_R = \{ R_1, R_2, \dots, R_n \}$ 。假设串行结构流水线包含 k 个任务,可表示为 $T = \{ t_1, t_2, \dots, t_k \}$ 。指派一个任务包括将其指派到哪个计算资源,以及分得多少个处理机,前者可定义为函数 A ,即

$$A(t_i) = R_j, t_i \in T, R_j \in G_R, \{ R_{dummy} \} \quad (1)$$

当还未给 t_i 指派资源时, $A(t_i) = R_{dummy}$, R_{dummy} 为哑元。假设 $A(t_i) = R_j$,一般地, t_i 的通信成本与 R_j 中分配给 t_i 的处理机个数无关,而 t_i 的计算成本却是处理机个数的函数。当多个流水线任务被指派到同一并行系统时,确定每个任务分得处理机的个数可用文献[5]的算法,该算法在分配处理机(或结点)时,使并行任务的 makespan 最小化,这与流水线高吞吐率的目标是一致的。

假设 $f_i^{exec}(A)$ 表示 t_i 的计算成本, $f_{i-1}^{com}(A)$ 表示 t_i 到 t_{i+1} 的通信成本,则 t_i 的响应时间可表示为

$$f_i(A) = \begin{cases} f_i^{exec}(A) + f_{i-1}^{com}(A), & i = 1 \\ f_{i-1}^{com}(A) + f_i^{exec}(A) + f_{i-1}^{com}(A) & i = (1, k) \\ f_{i-1}^{com}(A) + f_i^{exec}(A), & i = k \end{cases} \quad (2)$$

对于 $\forall A(t_i) \in G_R, f_i^{exec}(A)$ 都有一个值。当 t_i 无法在 $A(t_i)$ 上运行(受最少处理机数的限制)时,令 $f_i^{exec}(A) = \dots$

如果 $A(t_{i-1}) = R_{dummy}$, 对于 $\forall A(t_i) \in G_R, f_{i-1}^{com}(A)$ 都有一个确定值;反之,对于 $\forall A(t_i) \in G_R, f_{i-1}^{com}(A)$ 可定义为

$$f_{i-1}^{com}(A) = \text{aver}_{A(t_{i-1}) \in G_R} \{ f_{i-1}^{com}(A \oplus A(t_i)) \} \quad (3)$$

式中: $A \oplus A(t_i)$ 表示更新的 A 在 t_i 处的定义; $\text{aver}()$ 是对集合取均值。 $f_{i-1}^{com}(A)$ 的定义与 $f_{i-1}^{com}(A)$ 的类似。流水线吞吐率 T 可近似计算,即

$$T = 1 / \max_{i=1}^k (f_i) \quad (4)$$

在网格中指派流水线的任务以取得最大的 T , 这显然是个 NP 问题。但是,可以寻找一个 $A^{near-opt}$, 以实现较大的 T 。

2 任务指派算法

这里给出网格中串行结构流水线的一种任务指派算法 X-max-min, 它是对传统 max-min 算法的扩展,表现在: 指派一个任务包括指派任务到哪个并行系统,以及为任务分配多少个处理机; 任务都是可并行化的,在一个并行系统中的计算成本不是固定值,而是处理机个数的函数; 任务响应时间中包含通信成本,通信成本依赖于其他任务的指派; 流水线中任务并行,如果不止一个任务被指派到同一个并行系统,每个任务分得处理机的个数由文献[5]的算法确定; 通常一个任务的响应时间在指派完所有任务后才能确定。X-max-min 算法的形式描述如图 1 所示。

```

Input:  $f_i^{exec}(A), i \in [1, k], \forall \text{cpuT}(t_i) \in [1, p], p = \text{cpuR}(A(t_i))$ 
 $f_{i-1}^{com}(A), i \in [1, k], \forall A(t_i) \in G_R, \forall A(t_{i+1}) \in G_R$ 
Output:  $A^{near-opt}$ 
1. initialize(A); count = k;
2. while (count > 0) {
     $f_i(A) = \max\text{-min}(\{f_j(A) | A(t_j) = R_{dummy}, \forall t_j \in T\})$ ;
     $A = A \oplus A(t_i)$ ;
    update( $\{t_j(A) | A(t_j) = R_{dummy}, \forall t_j \in T\}$ );
    count = count - 1;
}

```

图 1 X-max-min 算法示意图

在图 1 所示的 Input 中, $\text{cpuR}(A(t_i))$ 是 t_i 所在并行系统中处理机的总数, $\text{cpuT}(t_i)$ 是 t_i 分得处理机的个数, $f_i^{exec}(A)$ 是 t_i 分得任意个数 ($[1, \text{cpuR}(A(t_i))]$) 处理机时的计算成本。 $\text{initialize}(A)$ 使得 $A(t_i) = R_{dummy}, \{f_i(A) | A(t_j) = R_{dummy}, \forall t_j \in T\}$ 包含所有尚未被指派的任务的响应时间函数,每个函数有其最小值, $\max\text{-min}()$ 返回这些最小值中的最大者,假设这个值是 $f_i(A)$ 在 $A(t_i)$ 处取得的运行成本,则用 $f_i(A \oplus A(t_i))$ 表示。 $\text{update}()$ 根据新的 A 更新未被指派的任务的响应时间函数。

文献[5]中 $\text{map-tasks}()$ 的每个任务指派是根据所有未被指派的任务被指派到(任选的)同一个处理机时的结果作出的,而本文 $\max\text{-min}()$ 确定的一个

任务指派是根据所有未被指派的任务被指派到各并行系统时的结果作出的,故 $\max\text{-min}(\)$ 是一种全局优化, $\text{map}\text{-tasks}(\)$ 是一种局部优化.

$f_i^{\text{exec}}(A)$ 的复杂度为 $O(nplbp)$, 文献[5]中算法的复杂度为 $O(p1bp)$, 其中 p 是机群结点个数的最大值, $f_{i-1}^{\text{com}}(A)$ 和 $f_{i+1}^{\text{com}}(A)$ 的复杂度都为 $O(n^2)$, 故 $f_i(A)$ 的复杂度是 $O(n^5plbp)$. 最差情况下, $\max\text{-min}(\)$ 是从 nk 个函数值中寻找一个值, 故其复杂度为 $O(n^6plbp)$. 图 1 算法的复杂度为 $O(k^5n^6plbp)$. 虽然 $f_i(A^{\text{nearopt}})$ 不等同于 t_i 被指派时的 $f_i(A)$, 但下面的实验表明 X-max-min 仍可使流水线取得较高的吞吐率.

3 仿真实验

网格实验床^[6]包含 3 个机群, RS6000 机群有 4 个 POWER3 结点, Sun 机群有 4 个 SPARC 结点, Intel 机群有 2 个 P4 结点. 假设 RS6000、Sun 和 Intel 机群分别用 R_1 、 R_2 和 R_3 表示, 则 $G_R = \{R_1, R_2, R_3\}$.

NCB3.0^[3]中的长流水线(LP)是个流水线应用, 但每个任务不是多处理机上的并行程序, 故不能用来验证 X-max-min 算法的有效性. 这里构造一个仿真流水线(eLP), 如图 2 所示.

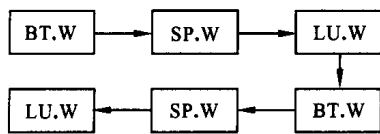


图 2 一个仿真流水线的 $DFG^{(3)}$ 框图

图中的任务来自 NPB2.3^[7], 按在流水线中的位置, 任务分别记为 t_1, t_2, t_3, t_4, t_5 和 t_6 , 各任务的计算成本见表 1.

表 1 任务的计算成本

| $A(t_i)$ | 计算成本 /s | | | | | | | |
|--|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| $\text{cpuT}(t_i)$ | $(R_1,1)$ | $(R_1,2)$ | $(R_1,4)$ | $(R_2,1)$ | $(R_2,2)$ | $(R_2,4)$ | $(R_3,1)$ | $(R_3,2)$ |
| $f_1^{\text{exec}}, f_4^{\text{exec}}$ | 36.8 | - | 15.1 | 272.3 | - | 156.4 | 36.0 | - |
| $f_2^{\text{exec}}, f_5^{\text{exec}}$ | 87.2 | - | 50.7 | 691.2 | - | 503.3 | 111.2 | - |
| $f_3^{\text{exec}}, f_6^{\text{exec}}$ | 46.5 | 40.9 | 20.9 | 539.8 | 335.7 | 220.9 | 64.2 | 39.3 |

注: - 表示 t_i 不能在此个数的处理机上并行运行.

网格中的网络连接可表示为 $\{L_{R_i-R_j} | R_i, R_j \in G_R\}$. $i \neq j$ 时, $L_{R_i-R_j}$ 表示并行系统之间的网络连接; 反之, 则表示并行系统内处理机间的网络连接. 表 2 是各网络连接的带宽(由 NWS^[8]测得).

t_i 到 t_{i+1} 的通信需求用 $c_{i(i+1)}$ 表示, 扩展式(1)中 A 的定义, 使 $A(c_{i(i+1)})$ 返回 $c_{i(i+1)}$ 被指派的

表 2 各网络连接的带宽

| 带宽 / $\text{Mb} \cdot \text{s}^{-1}$ | | | | | |
|--------------------------------------|---------------|---------------|---------------|---------------|---------------|
| $L_{R_1-R_1}$ | $L_{R_2-R_2}$ | $L_{R_3-R_3}$ | $L_{R_1-R_2}$ | $L_{R_1-R_3}$ | $L_{R_2-R_3}$ |
| 170.0 | 8.8 | 90.7 | 7.2 | 7.3 | 4.4 |

网络连接. 假设 $A(c_{i(i+1)}) = L_{R_1-R_1}(i \in [1,5])$ 时的 $f_{i(i+1)}^{\text{com}}(A)$, 再根据表 2(按通信成本与带宽成反比)计算 $A(c_{i(i+1)}) = L_{R_1-R_1}$ 时的 $f_{i(i+1)}^{\text{com}}$, 见表 3.

表 3 任务间的通信成本

| $A(c_{i(i+1)})$ | 通信成本/s | | | | | |
|--|---------------|---------------|---------------|---------------|---------------|---------------|
| | $L_{R_1-R_1}$ | $L_{R_2-R_2}$ | $L_{R_3-R_3}$ | $L_{R_1-R_2}$ | $L_{R_1-R_3}$ | $L_{R_2-R_3}$ |
| $f_{1,2}^{\text{com}}, f_{4,5}^{\text{com}}$ | 3.0 | 58.0 | 5.6 | 70.8 | 69.9 | 116.0 |
| $f_{2,3}^{\text{com}}, f_{5,6}^{\text{com}}$ | 2.0 | 38.6 | 3.7 | 47.2 | 46.6 | 77.3 |
| $f_{3,4}^{\text{com}}$ | 1.0 | 19.3 | 1.9 | 23.6 | 23.3 | 38.6 |

由 X-max-min 算法可得 A^{nearopt} , 表 4 是 A^{nearopt} 及当时的 $f_i^{\text{exec}}(i \in [1,6])$ 和 $f_i(i \in [1,6])$. 按式(4)计算 A^{nearopt} 时的 T 为 0.006 29 数据集/s.

表 4 由 X-max-min 得出的次优任务指派

| | t_1 | t_2 | t_3 | t_4 | t_5 | t_6 |
|-----------------------|-------|-------|-------|-------|-------|-------|
| A^{nearopt} | R_3 | R_1 | R_1 | R_3 | R_1 | R_1 |
| f_i^{exec}/s | 36.0 | 87.2 | 46.5 | 36.0 | 87.2 | 46.5 |
| f_i/s | 106 | 159 | 71.8 | 129 | 159 | 48.5 |

在 A^{nearopt} 下, $t_i(i \in [1,6])$ 自身不是并行的, 这是 R_1 和 R_3 中结点太少造成的. 假设 eLP 中每个任务表示实际的一个任务, 任务集合 V_G 为 $\{t_1, t_2, t_3, t_4, t_5, t_6\}$, 处理机集合 Q 为 $\{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}\}$, $\{q_1, q_2, q_3, q_4\}$ 是 R_1 中的处理机, $\{q_5, q_6, q_7, q_8\}$ 是 R_2 中的处理机, $\{q_9, q_{10}\}$ 是 R_3 中的处理机. 用文献[4]中的 $\text{map}\text{-tasks}(\)$ 可得任务指派 m 为 $\{t_1/q_4, t_2/q_6, t_3/q_3, t_4/q_2, t_5/q_5, t_6/q_1\}$, t_i/q_j 表示任务 t_i 被指派到处理机 q_j , 按式(4)计算此时的 T 为 0.001 36 数据集/s. 假设加速因子为 0.75, 用文献[4]中的 $\text{improve}(m)$ 可得改进的任务指派 m 为 $\{t_1/q_4, t_2/q_1, t_3/q_3, t_4/q_2, t_5/q_2, t_6/q_1\}$, 按式(4)计算此时的 T 为 0.007 21 数据集/s.

文献[4]算法中第一次调用 $\text{map}\text{-tasks}(\)$ 并未产生很高的 T , 这是因为 $\text{map}\text{-tasks}(\)$ 每确定一个任务指派时仅考虑各任务被指派到同一主机的情况, 而不是各任务被指派到各主机的全部情况. 本文算法得到的 T 略逊于文献[4]中最终的任务指派的 T , 这是因为本文算法不允许一个结点参加流水线中多个任务的执行.

(下转第 438 页)

$$(B - A)(I - B^*A)^{-1} < 1 \quad (7)$$

$$(B^* - A^*)(B - A) < (I - A^*B)(I - B^*A) \quad (8)$$

若令 $B = I$, 就得到推论 1.

推论 1 设 $A \in B(H)$, $\|A\| < 1$, $C \in B(H)$, $\|C\| < 1$, 则成立下述两个等价不等式

$$(A - C)(I - \bar{C}A)^{-1} < 1 \quad (9)$$

$$(A^* - \bar{C})^*(A - C) < (I - A^*C)(I - \bar{C}A) \quad (10)$$

下面, 给出复分析中推广的 Schwarz 引理的算子形式, 即定理 3.

定理 3 设 A 是 Hilbert 空间 H 上的真压缩算子, $f \in H(\mathbb{D})$, $|f(z)| < 1 (z \in \mathbb{D})$, 使得 $f(0) = 0$ ($\|A\| < 1$), 则

$$\tilde{f}(A)^* \tilde{f}(A) \leq (I - A^*)^{-1} (A^* - \bar{D}) (A - D) (I - \bar{A})^{-1} \quad (11)$$

$$\tilde{f}(A) \leq (A - D) (I - \bar{A})^{-1} \quad (12)$$

式(11)成为严格不等式当且仅当 $A \neq I$, 而且 f 不取 $\frac{z - \alpha}{1 - \bar{\alpha}z}$, $C \in B(H)$, $\|C\| = 1$.

以下, 我们将复分析中的 Dieudonne 的结果转化成算子形式, 得到定理 4.

定理 4 设 $f \in H(\mathbb{D})$, $|f(z)| < 1 (z \in \mathbb{D})$, $f(0) = 0$, $f'(0) = C$, 则算子集 $\{\tilde{f}(A) : A \in B(H), \|A\| < 1\}$ 是 $B(H)$ 关于零算子的星形集. 即对任何 $A \in B(H)$, $\|A\| < 1$, 任何 $t, 0 < t < 1$, 存在惟一的 $B \in B(H)$, $\|B\| < 1$, 使得 $\tilde{f}(B) = t\tilde{f}(A)$.

参考文献:

- [1] Ky Fan. Analytic functions of a proper contraction [J]. Math Z, 1978, 160(2) :275 ~ 290.
- [2] Rudin W. Functional analysis[M]. New York: McGraw-Hill, 1973.
- [3] Rudin W. Real and complex analysis[M]. New York: McGraw-Hill, 1966.
- [4] Halmos P R. A Hilbert space problem book[M]. New York: Springer-Verlag, 1980.
- [5] Elkinani A. Holomorphic functions in Hermitian Banach algebras[J]. Proc Amer Math Soc, 1991, 111(6) :931 ~ 939.

(编辑 杜秀杰)

(上接第 419 页)

4 结 论

目前, 网格中流水式计算的任务指派还是一个未解决的难题. 本文假设网格资源是并行系统的集合, 指派一个流水线任务包括指派到哪个并行系统以及分得其中多少个处理机. 并行系统中为多任务分配处理机的问题可以按最小化任务计算成本来解决. 任务响应时间中需要引入均值估计, 因为任务通信成本依赖其他相关任务的指派. X-max-min 是本文提出的任务指派算法, 它以任务响应时间函数为基础, 可取得较高的流水线吞吐率. 仿真实验证明了该算法的有效性.

对 X-max-min 算法得出的任务指派进行优化是必要的. 可重复改派响应时间最大的任务, 直到吞吐率不再增加为止. 通信成本显著时, 还可利用任务分群, 尽可能将通信需求大的一组任务指派到同一并行系统. 具体的优化方法将在以后研究.

参考文献:

- [1] Choudhary A N, Narahari B, Nicol D M. Optimal processor assignment for a class of pipelined computations [J]. IEEE Trans on Parallel and Distributed Systems, 1994, 5(4) :439 ~ 445.
- [2] Subhlok J, Vondran G. Optimal mapping of sequences of dar

ta parallel tasks [A]. 5th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Santa Barbara, USA, 1995.

- [3] Frumkin M, van der Wijngaart R F. NAS grid benchmarks: A tool for grid space exploration [A]. 10th IEEE International Symposium on High Performance Distributed Computing, San Francisco, USA, 2001.
- [4] Taura K, Chien A. A heuristic algorithm for mapping communicating tasks on heterogeneous resources [A]. 9th Heterogeneous Computing Workshop, Cancun, Mexico, 2000.
- [5] Krishnamurti R, Ma Y E. The processor partitioning problem in special-purpose partitionable systems [A]. IEEE Conference on Parallel Processing, University Park, PA, USA, 1988.
- [6] 桂小林, 钱德沛, 何戈. 基于校园网络的元计算实验系统 WADE 的设计与实现 [J]. 计算机研究与发展, 2002, 39(7) :888 ~ 894.
- [7] NASA Advanced Supercomputing. NAS Parallel Benchmarks [EB/OL]. <http://www.nas.nasa.gov/Software/NPB>, 2002 - 11 - 19.
- [8] Wolski R, Spring N, Hayes J. The network weather service: A distributed resource performance forecasting service for metacomputing [J]. Journal of Future Generation Computing Systems, 1999, 15(5/6) :757 ~ 776.

(编辑 苗 凌)

